The Design of a Novel E-cash System with the Fairness Property and Its Implementation in Wireless Communications

Chih-Hung Wang* and Wei-Ming Chiang

Department of Computer Science and Information Engineering

National Chiayi University

Chiayi 600, Taiwan

wangch@mail.ncyu.edu.tw

Received 30 March 2007; Revised 23 April 2007 ; Accepted 29 May 2007

Abstract. With the popularization of the networking, the network payments have become a trend. Electronic cash (e-cash) is such kind of payment; however, although various types of the demands in e-cash implementations have been conceived of, an e-cash scheme with fairness property has not yet been proposed. In this article, we present a novel electronic cash model, which uses a verifiable encryption and a restrictive blind signature to construct an electronic cash system with fairness between customers and merchants. The proposed e-cash scheme properly combines the payment protocol with a fair exchange procedure, so that the customer and the merchant can fairly exchange their money and goods. Our scheme can maintain fairness with the aid of an off-line trusted third party (off-line TTP). That means in a normal case, the customer and the merchant can receive their desired items without TTP's participation. However, only when a dispute occurs, the TTP can help both parties resolve the problem and ensure the fairness of the transaction. Moreover, this system can also protect customer's privacy, and prevent some hostile deceptions such as double spending and stealing. The system has also been implemented in a PDA emulator for wireless communications to evaluate the efficiency of running the program on a low computing power device.

Keywords: electronic cash, fair exchange, verifiable encryption, electronic commerce, wireless communications.

1 Introduction

Electronic cash systems have been widely discussed in past years in regards to many significant issues, such as anonymity, untraceability, unforgeability, unreuseability, non-repudiation and so on. Nevertheless, there still exist some problems between customers and merchants when an e-cash system is implemented on the Internet. For example, a customer who has paid electronic cash to a merchant cannot ensure that the merchant will send the goods which he has paid for. Similarly, a merchant who has sent goods to a customer also cannot ensure that the customer will send payment. To solve these problems, we integrate fair exchange with the payment protocol of our e-cash system.

There are two different types of electronic cash systems: on-line and off-line. In an on-line e-cash system, the issuing bank should participate in the payment protocol to verify the coin. This may be a straightforward way to make sure of the validity of payments, but it is inefficient for real-time transactions. An off-line system can enhance performance in which the bank is not required to be present to verify the coin during the payment procedure; however, a double-spending detection mechanism must be properly designed.

Electronic cash was introduced by Chaum [14]. Chaum used a blind signature to provide "anonymity" and "untraceablility" for electronic cash. Afterwards, Brands in 1993 provides an untraceable electronic cash scheme in wallet with observers [6]. In his paper, he proposed that a coin can be traced only if double-spending occurs. Song and Korba [26] described how to construct an electronic cash scheme which has both features of "untraceability" and "non-repudiation". This means it can be ensured that a legal user would never be traced; however, once a dispute happens, a user cannot deny that he had spent the electronic cash.

Many variants of e-cash scheme have been proposed for different applications. Okamoto [23] proposed a "divisible" electronic cash scheme, which expresses the amount of money by a tree structure to divide the electronic cash into a minimum unit. In the divisible electronic cash scheme, any remainder money can be continually used [22]. Kim et al. proposed a "fair tracing" protocol to protect customers from suffering illegal tracing by a bank or other parties [19]. Camenish et al. proposed a scheme in which once a user spends one of coins in his wallet

* Correspondence author

twice, all coins in his wallet can be traced [9]. Hou and Tan eliminated the withdrawal phase of the electronic cash system. Fair traceability was also provided in their scheme in case of crimes taking place [18].

There are still more papers that consider efficiency. Chen et al. proposed a scheme that uses a proxy to reduce the burden of the merchant [10]. Chan et al. bounded each procedure in their divisible cash scheme by tens of exponentiations. [8]. Camenish et al. proposed a scheme with lower complexity of wallet size [9].

As we see, most of electronic cash schemes have not considered the "fairness" between customer and merchant. An on-line transaction is not a face-to-face service; for this reason, a dishonest buyer or seller may destroy the fairness of the transaction. An exchange is fair if at the end of exchange, each party receives the expected item or neither party receives any useful information about the other's item. The last two decades of research have given us useful information on solutions to the fair exchange problem. Solutions to the fair exchange problem reported in past literature fall into the following two categories: (1) Gradual exchange protocols: two parties gradually disclose the expected items by many steps [4][24]. (2) Trusted Third party protocols: two parties exchange their expected items by the aid of on-line or off-line trusted third party [1][2][5][3].

Purposes and contributions. We have constructed a fair electronic cash system that can protect a user's privacy and maintain fairness during the transactions. Previous fair payment systems have been based on signature exchange and will reveal a user's identity. In our scheme, even when a dispute occurs, the buyer's identity will not be revealed to the bank/TTP.

Organization. The rest of this article is organized as follows: In Section 2, we briefly introduce some techniques that are used to realize our protocol. In Section 3, we describe the basic model of our fair electronic cash scheme, and detailed procedures are proposed in Section 4. The security and fairness analysis is presented in Section 5. Then we demonstrate the implementation in wireless communications and discuss its efficiency in Section 6. Finally, the concluding remarks are given in Section 7.

2 Preliminaries

2.1 The Discrete Logarithm Assumption

The Discrete Logarithm Assumption (DLA) is an assumption that solving the discrete logarithm problem is believed to be difficult. The discrete logarithm problem is defined as follows:

Given an element g in a group G of order q and another element y of G, find x where $0 < x < q$-1 such that $y = g^x$. It is computationally hard to get x from y.

2.2 Double Exponentiation and Double Discrete Logarithm

Let q, q' be two large primes where $q' \mid q-1$ and $f \in Z_q^*$ be an element of order q'. It is computational difficult to find a discrete logarithm to the base g and to the base f. Double exponentiation with base g and f is defined as: $Z_{q'} \to G : x \mapsto g^{(f^x)}$.

By the double discrete logarithm of $y \in G$ to the base g and f, when $y = g^{(f^x)}$, there is an unique $x \in Z_{q'}$ if it exists.

2.3 Verifiable Encryption of Discrete Logarithm

In our scheme, a verifiable encryption is needed, so we briefly introduce one [27]. The encryption is identical to the ElGamal scheme, which is a variation of the Diffie-Hellman key-exchange protocol.

In the verifiable encryption method, each participant chooses a secrete key $x \in Z_{q'}$, and publishes his public-key $y = f^x \pmod q$. The dealer randomly chooses $\alpha \in Z_{q'}$ to encrypt a message $m \in Z_q^*$ with his public-key y. The encrypted message is a pair of $(C_1, C_2) = (f^\alpha, m^{-1} y^\alpha) \pmod q$. The cipher-text (C_1, C_2) can be decrypted as: $m = C_1^x / C_2 \pmod q$.

Here is a protocol for verifying that a pair (C_1, C_2) is a ciphertext for the logarithm of a public element $M = g^m$. Since the equation $M^{C_2} = g^{mC_2} = g^{(y^{\alpha})}$ holds, one can perform the following protocol to prove to the verifier that the discrete logarithm of C_1 to the base f is equal to the double discrete logarithm of M^{C_2} to the base g and y. The interactive proof proposed in [27] is shown in Fig. 1.

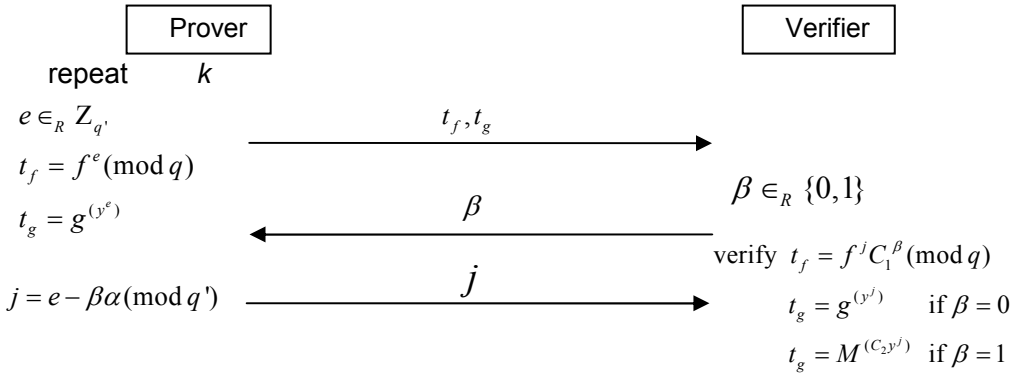

Fig. 1. An iterative proof of the verifiable encryption

Stadler suggested a non-interactive proof for the verifiable encryption [27]. Let $H_l : \{0,1\}^* \to \{0,1\}^l$ be a hash function ($l \approx 100$). The prover chooses $e_i \in_R Z_{q'}$ and computes $t_{f_i} = f^{e_i} \pmod q$ and $t_{g_i} = g^{(y^{e_i})} \pmod q$, for $i = 1...l$. He then computes $R = (j_1, ..., j_l) = (e_1 - \beta_1\alpha \pmod{q'}, ..., e_l - \beta_l\alpha \pmod{q'})$, where β_i is the i-th bit of $\beta = H_l(M \| C_1 \| C_2 \| t_{f_1} \| t_{g_1} \| ... \| t_{f_l} \| t_{g_l})$.

The prover sends R and β to the verifier and then the verifier computes $t_{f_i} = f^{j_i} C_1^{\beta_i} \pmod q$ and $t_{g_i} = (g^{1-\beta_i} M^{\beta_i C_2})^{(y^{j_i})}$ for $i = 1...l$. The verifier accepts the proof as valid if he checks that β holds true.

3 The Basic Model of the System

In this section, we briefly describe our fair electronic cash scheme. There are four participants in this scheme: the Bank/TTP, the customer, the merchant, and the registration center, which can be written as B/T, C, M, RC respectively. The bank here also plays the role of resolving the dispute. Thus, the key pairs for the bank to sign the electronic cash and for the TTP to encrypt the goods are different. We assume that B/T should be trusted; i.e., B/T will not maliciously reveal the knowledge that he knows to other users. The customer can withdraw electronic cash from B/T before he goes shopping on Internet. He wants to protect his privacy in the withdrawal, payment, and deposit protocols, and hopes to have a fair payment with the merchant. The merchant sells his soft goods on Internet. He must register all his goods to RC before he sells them. The registration center is responsible for verifying whether the goods (e.g. a serial number) are valid or not, and generates a signature on the description of goods as a certificate. Here we assume that the registration center is trusted enough that he will not reveal the information regarding the goods to any other party.

There are four phases in the proposed protocol (see Fig. 2):

Initial Phase: The initial phase includes the following three subprocedures. (1) Key-pair Setup: setup user's public / private key, bank's public / private key, and TTP's public / private key. (2) Account Establishment: the users' (including customer and merchant) accounts in the bank are needed to be established before starting a transaction. (3) Goods Registration: all merchandise here can be regarded as digital data, for example, serial numbers, passwords, etc. They must be registered by the registration center.

Withdrawal Phase: The customer in this phase can acquire electronic cash from the bank by running the blind signature. The withdrawal phase is required to be designed to protect customer's privacy.

Payment Phase: In this phase, customer and merchant aim to exchange their money and goods. We combine the payment procedure with the fair exchange protocol whereby both the customer and the merchant can get what they want or neither of them can receive useful information. When a dispute occurs, some steps may not execute normally in the payment phase. Thus, customer or merchant can ask for TTP's help.

Deposit Phase: The merchant can change his coins received from the customer into real money via the bank. The detailed procedures are presented in next section.

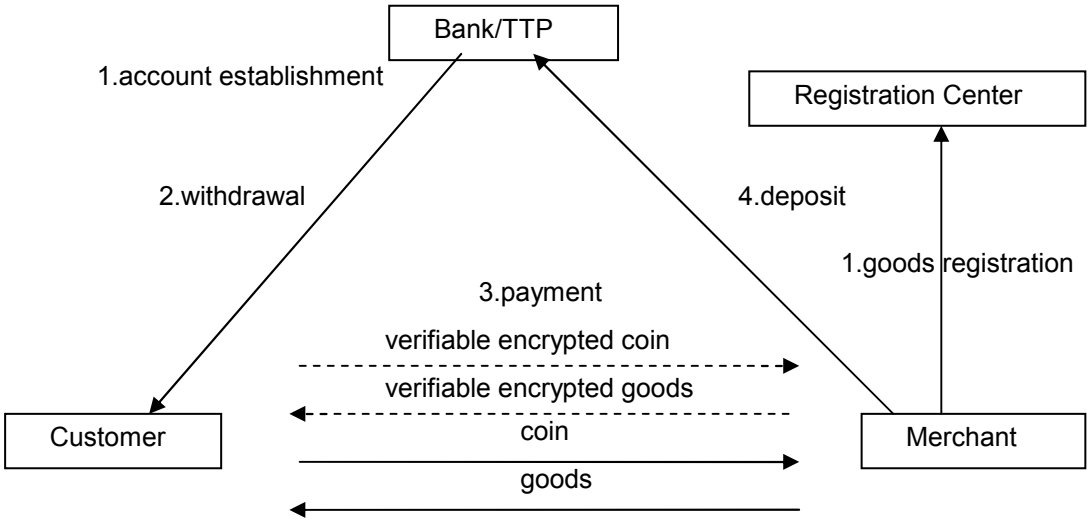

Fig. 2. Fair e-cash scheme with off-line TTP (normal)

4 The Proposed E-cash Scheme

The Initial Phase: Assume that q and q' are defined as before. Let $g, g_1, g_2 \in G$ be three generators of G_q of order q. B/T has two key pairs. When he plays as a bank to sign the electronic cash for the customer, his private key is $x' \in Z_{q'}$, and the corresponding public key is $h = g^{x'}$. When he plays as a TTP to encrypt or decrypt the electronic cash, his private key is $x \in Z_{q'}$ and the corresponding public key is $y = f^x \bmod q$. C chooses a random number $u_1 \in Z_q$ and computes $I = g_1^{u_1}$ as his account information (note that $g_1^{u_1} g_2 \neq 1$), and then he sends I to B. Next, B computes $z = (Ig_2)^{x'}$ and sends it to C.

M computes all $PI_{goods_i} = g^{goods_i} \bmod p$, for i=1,2,…,n, where the number of n denotes the amount of the goods. M then sends his $ID = I_M$, Public Information =PI_{goods}, goods, and the description to RC. RC then verifies the goods and the description. If they are correct, RC signs them. The transmissions during this phase are running through a secure channel.

The Withdrawal Phase: This phase is similar to that of [6]. At the end of the protocol, C can get $coin = A, B, (z', a', b', r')$, where (z', a', b', r') is the signature on (A, B). The additional steps are that C uses TTP's public key to encrypt a part of the coin (i.e. r'). The result is $coin' = (A, B, z', a', b', (C_1, C_2) = (f^\alpha, r'^{-1} \cdot y^\alpha))$, where y is TTP's public key, and α is a random number selected by C.

The Payment Phase: There are four steps in the payment phase (see Fig. 3). In Step 1, C sends $coin' = (A, B, z'a', b', (C_1, C_2))$ to M. If $A \neq 1$, M computes $d = H_0(A, B, I_M, date/time, desc_{goods_i})$ and sends it to C. Then C computes $r_1 = d(u_1 s) + x_1 \ (\bmod q)$, $r_2 = ds + x_2 (\bmod q)$, $R' = g^{r'}, A' = A^{r'}$, and sends r_1, r_2, R', A' to

M. Here, M can make sure that C is the owner of this coin by verifying $g_1^{r_1} g_2^{r_2} = A^d B$ and the coin is correct by verifying $R' = h^{c'} a'$ and $A' = z'^{c'} b'$.

Further notice that M does not know the value of r' in the coin. Thus, C must prove to M that he knows the correct value of r' and what he sends to M includes an encrypted r'. To reduce the computation costs, we apply a non-interactive proof proposed in [27] (see Fig. 4).

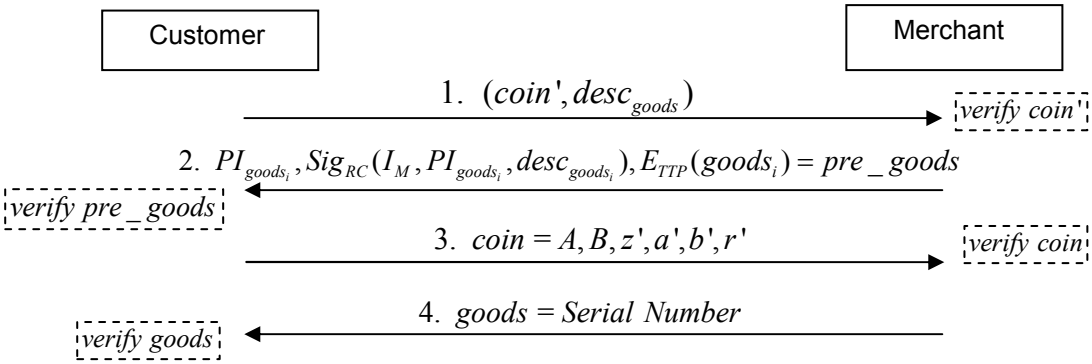

Fig. 3. Four steps in payment phase

After M verifies the $coin'$ successfully, M sends the certificate of pre_goods to C. Then M makes a non-interactive proof to C to convince C that the $goods_i$ is encrypted by TTP. If the proof is valid, C is convinced that pre_goods can be decrypted to $goods_i$ which he wants to buy.

In Step 3, C sends a real coin to M. After verifying the coin, M sends $goods_i$ to C in Step 4. If C sends his $coin'$ to M but does not receive the pre_goods after a period of time, he would query M. If the answer is "yes", that means M had received $coin'$. Then C will wait for next time period. If C still does not receive the goods, he will run Cancel phase.

The Cancel Phase: This sub-protocol is for C to cancel the transaction. C sends (A, B, z', a', b', r'), $r_1, r_2, date/time, desc_{goods_i}$ to B/T. After verifying these messages, the bank stores them in the Database_C. A coin which stored in Database_C is not allowed to be deposited by M in the future.

The Resolve_M Phase: After M sends pre_goods to C, he can run this protocol if he did not receive the coin from C. M sends $A, B, z', a', b', (C_1, C_2) = (g_1^{\alpha}, r^{-1} \cdot y^{\alpha}), R', A', r_1, r_2, date/time$, $desc_{goods_i}, pre\_goods_i$ to B/T, B/T will check whether they are in the Database_C. If not, TTP decrypts $coin'$ and sends the real coin to M and the goods to C.

The Resolve_C Phase: If C sends the coin to M but does not receive the deserved goods, he can send $A, B, (z', a', b'r'), r_1, r_2, date/time, desc_{goods_i}, pre\_goods_i$ to B/T. B/T then verifies $g^{r'} = h^{c'} a'$, $A^{r'} = z'^{c'} b'$ and $g_1^{r_1} g_2^{r_2} = A^d B$ to confirm the correctness and the ownership of the coin. If the above verifications pass, B/T decrypts pre_goods_i and verifies $PI_{goods_i} = g^{goods_i}$. B/T then sends the goods to C and the real coin to M.

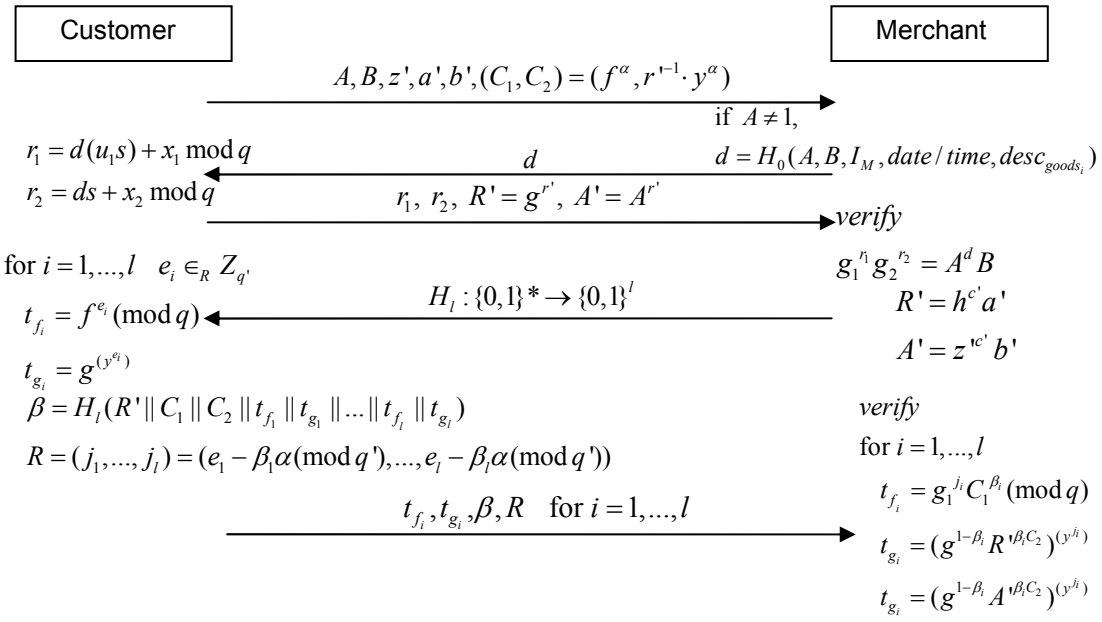

Fig. 4. The first step of the payment phase (verification of the pre_coin)

The Deposit Phase: If M finishes the payment protocol in the normal case or successfully runs the Resolve_M, he will obtain the coin= $A, B, (z', a', b', r')$. Now he can perform this protocol for deposit.

5 Security and Fairness Analysis

5.1 Security Analysis

Security of e-cash is defined in terms of six requirements: Unreusability, Untraceablity, Unforgeablility, Unexpandability, Unlinkability and Unstealability [17][21]:

Unreuseability: If a coin has been deposited in the bank twice, the bank would be assumed to receive two messages (d, r_1, r_2) and (d', r_1', r_2') which are produced during the payment phase. Thus, the bank can reveal $u_1 = g_1^{(r_1 - r_1')/(r_2 - r_2')}$ and compute the customer's identity $I = g_1^{u_1}$. This situation shows that once the user uses the coin twice, his account information will be traced by the bank.

Untraceability: This property can be proven under the security assumption of used blind signature which has been shown in [6]. The bank cannot link the identity of the customer and the payment coins, since c' and r' are unknown to the bank in the withdrawal phase. Another security assumption is based on a representation problem in group of prime order. That is, given a generator tuple $(g_1, ..., g_k)$ and $h \in G_p$, to find a representation of h with respect to $(g_1, ..., g_k)$ is hard. In this scheme, a customer's identity is defined as $I = g_1^{u_1}$, a coin is represented as $A, B, (z', a', b', r')$ where $A = (Ig_2)^s = (g_1^{u_1} g_2)^s$. Anyone who wants to trace the customer's identity must know a presentation of A with respect to g_1 and g_2. If the representation problem is hard to solve, it can be said that the customer's identity is untraceable.

Unforgeability: A legal coin should include the bank's signature, namely (z', a', b', r'), which can be verified by $g^{r'} = h^{H(A,B,z',a',b')} a'$. The value r' is computed by the equations of $r' = ru + v (\bmod q)$ and $r = cx' + w (\bmod q)$ where x' is bank's private key. Anyone who has no bank's private key cannot compute such a signature, and hence cannot forge a valid coin.

Unexpandability: Assume that there are only N coins which are made by N withdrawal protocols. If the property of unforgeability is established, it is infeasible to generate a different coin from the original N coins. When N+1 deposit protocols occur, there are two kinds of situation. One situation is that a customer spent his coin twice, and his identity was revealed by double spending checking ($g = (r_1 / r_1')/(r_2 / r_2')$). The other situation occurs when the merchant deposits the same coin twice. This kind of behavior can be easily detected by the bank to check whether there are the same r, r', $date/time$ and $goods_i$ in his database.

Unlinkability: Assume that two coins: $coin_1 = (A_1, B_1, (z_1', a_1', b_1', r_1'))$ and $coin_2 = (A_2, B_2, (z_2', a_2', b_2', r_2'))$ were withdrawn by the same customer, where $A_1 = (Ig_2)^{s_1}$, $B_1 = g_1^{x_{11}} g_2^{x_{21}}$ and $A_2 = (Ig_2)^{s_2}$, $B_2 = g_1^{x_{12}} g_2^{x_{22}}$. Because of the representation problem of the prime order, no one can determine the link between A_1 and A_2 without knowing the secret s_1, x_{11}, x_{21} and s_2, x_{12}, x_{22} those being kept by the owner.

Unstealability: During the payment protocol, the customer must prove to the merchant that he is the owner of the coin. Upon receiving $d = H_0 = (A, B, I_M, date/time)$, the customer sends r_1, r_2 to the merchant, where $r_1 = d(u_1 s) + x_1 \pmod{q'}$ and $r_2 = ds + x_2$. Here u_1, s, x_1, x_2 should be kept secret by the customer. Even if someone can steal a coin, he cannot spend this coin without proving that he is the owner of this coin.

5.2 Fairness Analysis

There are three cases in discussing the fairness of our protocol.

Case 1 (Both C and M behave properly): It is easy to see that in this case C obtains the goods and M obtains the coin.

Case 2 (M behaves improperly): There are four ways that M may behave improperly.

(1) M does not send pre_goods to C after he receives the $coin'$ from C: After waiting for a response from M for a short time (assume t ms), C will query M again. If there still no answer, C will run the Cancel sub-protocol to stop the transaction. The fairness can be maintained even if M performs the Resolve_M sub-protocol before C runs the Cancel sub-protocol, since B/T still needs to send the goods to C in the last step.

(2) M sends an invalid pre_goods to C: In the second step of the payment phase, M need to sends the following information to C: $PI_{goods_i}, Sig_{TA}(I_M, PI_{goods_i}, desc_{goods_i}), E_{TTP}(goods_i) = pre\_goods_i$. C first checks the signature $Sig_{TTP}(I_M, PI_{goods_i}, desc_{goods_i})$ with $desc_{goods_i}$. If $desc_{goods_i}$ is satisfied, C can use the public information PI_{goods_i} in the signature to verify the proof of the verifiable encryption made by M (i.e. check the equation $PI_{goods_i} = g^{goods_i} \bmod p$). C can verify the pre_goods (i.e. $E_{TTP}(goods_i)$) so that M cannot successfully make a proof for this invalid pre_goods to convince C.

(3) M receives the coin from C but refuse to send the goods to C: In this case, C already received a valid pre_goods and then he can ask B/T to decrypt the pre_goods (see Resolve_C). After doing Resolve_C sub-protocol, C can obtain the goods he desired.

(4) M receives the coin at Step3 in the payment protocol but sends invalid goods to C: The incorrect goods will not pass the verification of $PI_{goods_i} = g^{goods_i} \bmod p$. C can ask B/T to resolve the dispute, and obtain the valid goods by running Resolve_C sub-protoocl.

Case 3 (C behaves improperly): There are also four ways that C may behave improperly.

(1) C sends an invalid $coin'$ or wrong $desc_{good_{si}}$ to M: First, M checks whether $desc_{good_{si}}$ belongs to him or not. If yes, M checks the correctness of $coin'$ by verifying the zero-knowledge proof of verifiable encryption from C. If not, M reject the transaction. In our scheme, $coin' = (A, B, z', a, b, (C_1, C_2))$ where (C_1, C_2) is cipher of r'. C must send $R' = g^{r'}$ and $A' = A^{r'}$ to M. Since g and A is public to M, if C sends wrong R' or A' to M, the verification of the following two equations will not be satisfied: $R' = h^{c'} a'$, $A' = z'^{c'} b'$. Moreover, if C sends an invalid $coin'$ (i.e. (C_1, C_2)) to M, he cannot prove that he knows the element r'. An invalid $coin'$ cannot pass the verification and thus the fairness can still be maintained because the both two parties obtain nothing.

(2) After receiving pre_goods from M, C performs Cancel: There are two possibilities in this case. In the first case, C does not run Resolve_C then the both sides obtain nothing. The other case occurs when C runs Resolve_C then both sides obtain their desire information.

(3) After receiving pre_goods from M, C refuses to send coin to M: In this case, C has pre_goods and M has $coin'$. If one of them performs resolve procedure to ask help of B/T, they will both obtain their desired

items. If not, neither of them obtains useful information. In light of the fact that C can perform Resolve_C and M can perform Resolve_M, let us then consider what happens if both C and M perform their resolve sub-protocol. In this situation, C will receive two $goods_i$'s from B/T and M will receive two *coin*'s from B/T. Two $goods_i$'s have the same serial number that can be copied by a user easily. How to prevent a user from copying $goods_i$ and using it arbitrarily is not important here. The point is that M has two copies of *coin*, and can he deposit both of them? The answer is no. Because M cannot create another pair of r_1, r_2, called r_1', r_2', since he does not know u_1, x_1, x_2. If he uses the same r_1, r_2 to deposit the *coin* twice, the B/T can easily ascertain that it is a double deposit by M. So, even if each side performs a resolve sub-protocol and receives two copies, the fairness between C and M still holds.

(4) C sends invalid coin after he receives correct pre_gooods from M: If C sends incorrect *coin* or r' to M in Step 3 of the payment phase, the following verification equations will not pass: $g^{r'} = h^{c'} a'$, $A^{r'} = z'^{c'} b'$. If the verification is abortive, M can run Resolve_M to obtain his *coin*.

6 Implementations and Efficiency

6.1 Implementation Environment

To observe the efficiency and practicability of our scheme, we implemented it and evaluated its computing time. We first realize our scheme on PC. The experimental platform is the IntelTM Pentium 3.2 GHz PC, with 1536 MD DDR RAM and Java 2 SDK 1.4.x. Subsequently, we implemented our scheme on a personal digital assistant (PDA) to evaluate the efficiency of running on a low computing power device. We choose Nokia 9210c as our experimental platform because it can offer us many useful tools such as Nokia Symbian 6.0 SDK and EPOC Emulator.

The hardware resources of a PDA such as computing power and memory size are not as ample as that of a PC. Moreover, there are still several details that we should take care; for example, the useable memory size on a PDA device is limited to 8 MB, and classes that we can use on the PDA with Pjava are only supported before Java 1.1.

PJEE (Personal Java Emulation Environment) is a JRE (Java run time environment) developed on smaller devices like PDAs. After we install PJEE on our digital product (i.e. Nokia 9210c), we can run our programs written in Java language on it with only minor modifications. If we write the program with JDK whose version is over 1.1, we must compile the program with a proper vision and a proper class path. For example, the command can be written as:

C:\...>javac –classpath D:/pjee1.1\lib\classes.zip –target 1.1 Main.java

In addition, we packed our program into a .jar file which can be directly executed on the PDA.

6.2 System Implementation Architecture

Fig. 5 indicates the system implementation architecture. By using the library of java language, we can easily write some cryptographic functions, like big prime integer generation, primality testing, discrete logarithm, blind signature, message authentication code, etc.

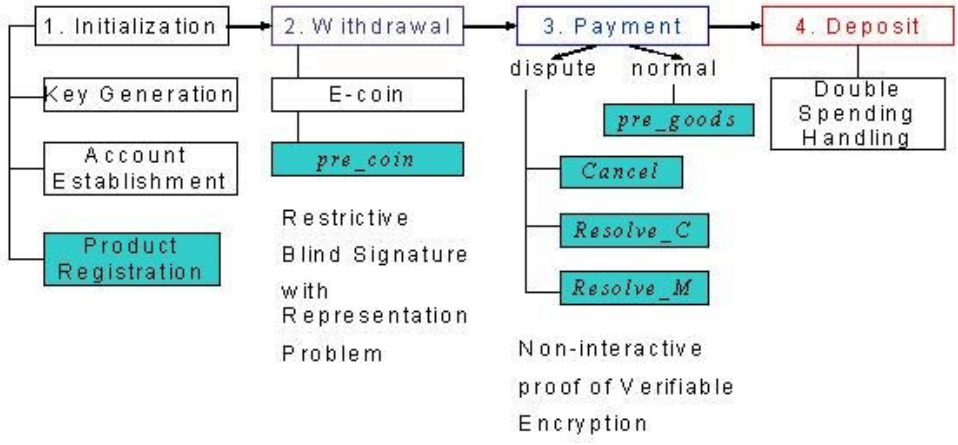

Fig. 5. The system implementation architecture

6.3 System Simulation on the Server Side

First, we use a double discrete logarithm program to produce the environmental parameters. The program produces a big prime number with length of 512 bits and 1024 bits called q', and then finds q and p such that $q = 2q'+1$ and $p = 2q+1$. If the program cannot find such q and p, it repicks a big prime number q', until the above conditions are satisfied. The program then finds out the generators f, g, g_1, and g_2. This is a time-consuming procedure (usually running more than 10 hours), and the speed depends on the probability. After we find out p and q, we store them for repeated usage.

The public keys and private keys for all participants (including customers and merchants) can be computed by using the parameters p and q which we figured out before. The following shows the equations:

User's key pair: $(u_1, I = g_1^{u_1})$

Bank's key pair: $(x', h = g^{x'})$

TTP's key pair: $(x, y = f^x)$

Fig. 6 shows the produced process of the above-mentioned key and the corresponding account information. Before the merchant starts to sell his goods, he must register the goods with the registration center.

6.4 System Simulation on the Client Side

In the withdrawal phase, the customer needs to withdraw electronic cash from the bank. With a blind signature technique, we need to compute $c = c'/u \bmod q$, where $c' = H(A, B, z', a', b')$. It may be worth to pointing out that Pjava did not support the library of the message authentication code (MAC). Hence, we need to create a class for it to run on the client side. We designed the MAC as a 160-bit output hash function just like SHA-1, which also can be used directly on the server side.

In payment phase, the customer needs to make a transaction with the merchant under the wireless network environment. Some parameters need to be produced in advance. Subsequently, the customer sends his pre_coin to the merchant and proves its correctness by a zero-knowledge proof. After this, the merchant sends his pre_goods to the customer and also proves its correctness. We implemented both iterative and non-iterative verifications. In the case of non-iteration verification, we can reduce the executing time for about $(2l-1+1)t_t$ ms, where t_t denotes the time of a network transmission and l denotes the number of repetitions in the iterative verification procedure. Assume that $t_t = 30$ and $l = 100$, the non-iterative way will reduce (2*100)*30 ms = 6 second. The process of the iterative verification is shown in Fig. 7.

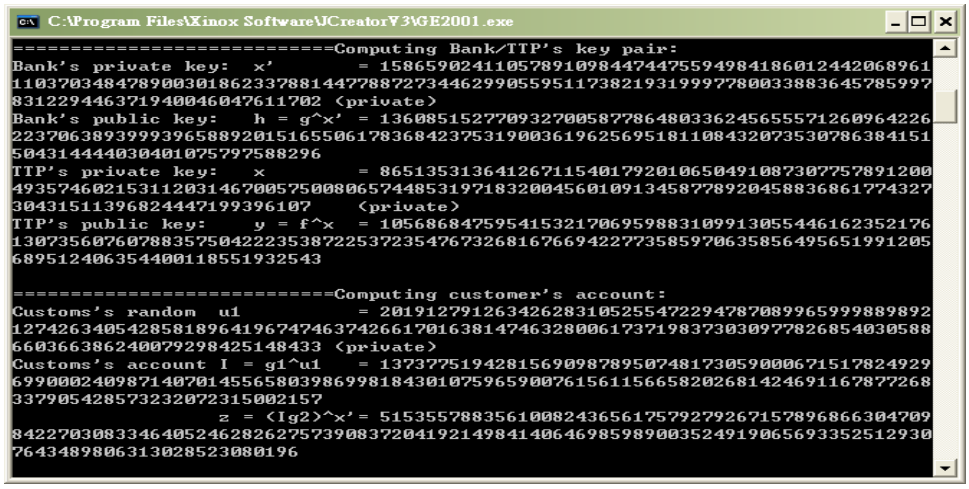

Fig. 6. The process of key generalization and account establishment

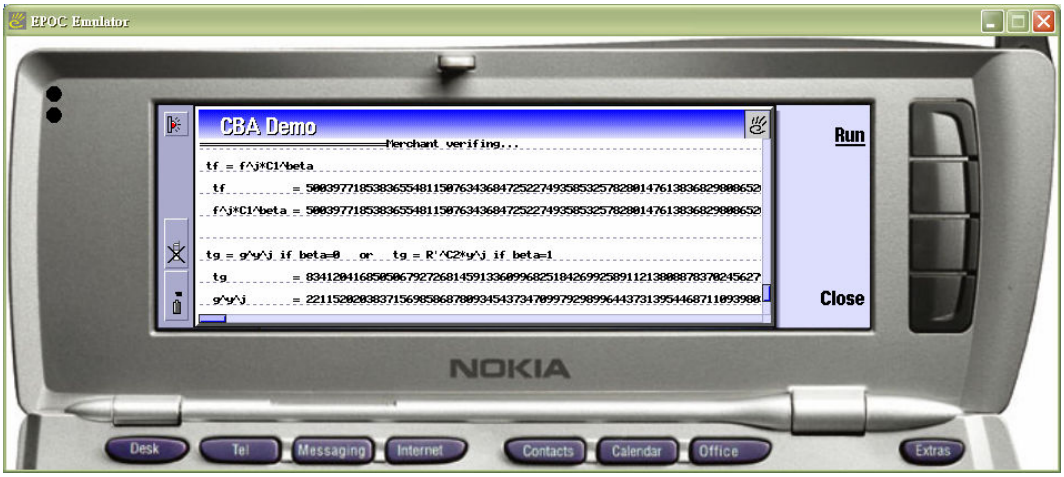

Fig. 7. The process of the iterative verification

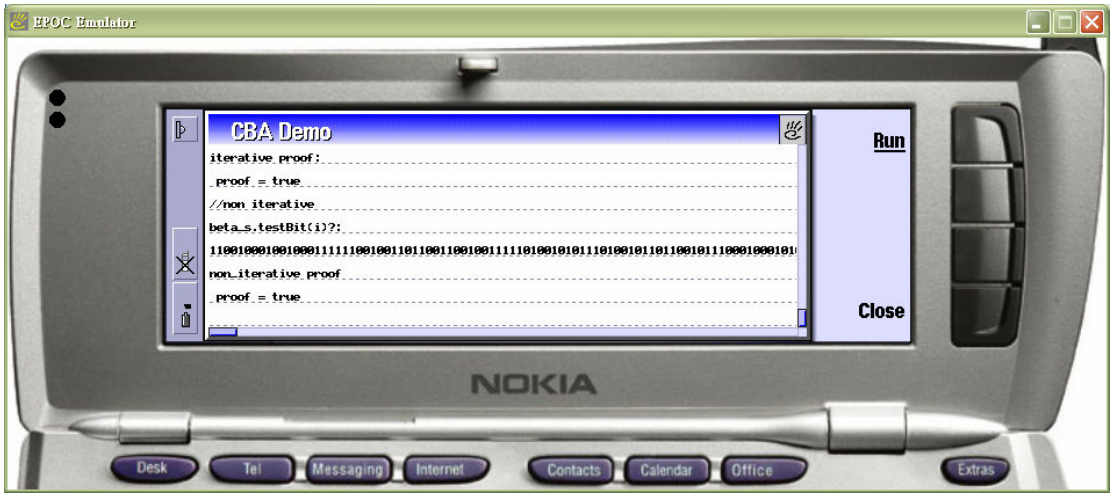

Fig. 8. The result of the verification

For the non-iterative verification, we need to compute n_t different values of e', t_{f_i}, t_{g_i} and compute $\beta = H_l(R' \| C_1 \| C_2 \| t_{f_1} \| t_{g_1} \| ... \| t_{f_l} \| t_{g_l})$ and $R = (e_1 - \beta_1 \alpha (\bmod q'), ..., e_l - \beta_l \alpha (\bmod q'))$, where R is a bit stream with l bits. Fig. 8 shows the result of the verification and the parameter R in the non-iterative verification.

6.5 Efficiency Analysis

To estimate the efficiency of this system, we first figure out the execution time of the initial phase, withdrawal phase and deposit phase. The result shows that each of them does not take more than one second. The most time-consuming procedure is placed in the payment phase, i.e. the zero-knowledge proof. In the client (customer) side, performing the proof of verifiable encryption for 100 rounds (security level 2^{-100}) with a 512-bits q' needs 9.8 seconds. Notably, the following experimental results of the client side are based on running the programs on an emulator instead of a real PDA device. Since a real PDA has more limitations on memory resources, the performance should be lower than what we list below.

Table 1 shows the time needed in the verification for the client side based on the different values of l. Small l and size of q' can be more suitable for wireless environment but will have less security. Time needed in the verification for the server side is shown in Table 2. From Table 2**,** we can see that the computing time greatly increases because of large number of rounds in the proof procedure. Even in the server side, we need about 4 seconds to perform the proof for 100 rounds.

Table 1. Time complexity in the client side (on Nokia 9210c EPOC emulator)

	l =10	l =50	l =100	l =160
Total Time of Client Side with *q'* of 512 bits	≅ 1.2	≅ 5.2	≅ 9.8	≅ 15.4
Total Time of Client Side with *q'* of 1024 bits	≅ 9.5	≅ 29.2	≅ 57.1	≅ 91.5

l: number of rounds (sec)

Table 2. Time complexity in the server side (on PC)

	l =10	l =50	l =100	l =160
Total Time of Server Side with *q'* of 512 bits	≅ 0.4537	≅ 2.2316	≅ 4.4365	≅ 7.0925
Total Time of Server Side with *q'* of 1024 bits	≅ 4.6528	≅ 17.8562	≅ 32.4126	≅ 49.5427

l: number of rounds (sec)

7 Conclusion and Future Works

In this article, we realize a fair electronic cash system by using a verifiable encryption in the payment phase. The customer can keep his privacy, and his identity will not be traced by anyone if he does not spend his coins twice. By using a non-interactive proof and three sub-protocols of fair exchange, the customer and the merchant can fairly exchange their coins and goods. Nobody can gain an advantage in the exchange.

But we have to say with regret that our scheme has high computational costs; we are investigating how to reduce the computation complexity in the future.

Acknowledgments

This work was supported in part by TWISC@NCKU, National Science Council under the Grants NSC 94-3114-P-006-001-Y.

References

[1] N. Asokan, M. Schunter and M. Waidner, "Optimistic protocols for fair exchange," *Proceed-ings of 4th ACM Conference on Computer and Communications Security*, pp.6-17, 1997.

[2] N. Asokan, V. Shoup and M. Waidner, "Asynchronous protocol for optimistic fair exchange," *Proceedings of 1998 IEEE Symposium on Security and Privacy*, pp.86-99, 1998.

[3] N. Asokan, V. Shoup and M. Waidner, "Optimistic fair exchange of digital signatures," *Proceedings of IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 4, pp. 593 -610, 2000.

[4] E. F. Brickell, D. Chaum, I. B. Damgard and J. van de Graaf, "Gradual and verifiable release of a secret," *Advances in Cryptology – Proceedings of Crypto '87*, pp.156-166, 1987.

[5] F. Bao, R. H. Deng and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP," *Proceedings of 1998 IEEE Symposium on Security and Privacy*, pp.77-85, 1998.

[6] S. Brands, "Untraceable off-line cash in wallets with observers," *In Advances in Cryptology – Proceedings of Crypto'93, Lecture Notes in Computer Science*, pp. 302-318, Springer-Verlag, 1994.

[7] D.Chaum, A. Fiat and M. Naor. "Untraceable electronic cash," *In Advances in Cryptology- Crypto'88, Lecture Notes in Computer Science*, pp. 319-327, Springer-Verlag, 1990.

[8] A. Chan, Y. Frankel and Y. Tsiounis, "East come-easy go divisible cash," *In Advances in Cryp-tology – Proceedings of Eurocrypt '98, Lecture Notes in Computer Science*, pp.561-575, Springer-Verlag, 1998.

[9] J. Camenisch, S. Hohenberger and A. Lysyanskaya, "Compact e-cash," *In Advances in Cryp-tology – Proceedings of Eurocrypt '05, Lecture Notes in computer Science*, pp. 302-321, Springer-Verlag, 2005.

[10] Y. Y. Chen, J. K. Jan and C. L. Chen, "A novel proxy deposit protocol for e-cash systems," *Proceedings of ELSEVIER Applied mathematics and Computation*, Vol.163, No.2, pp. 869-877, 2005.

[11] T. J. Ca, D. D. Lin, and R. Xue, "A randomized RSA-based partially blind signature scheme for electronic cash," *Computer & Security,* Vol.24, No.1, pp. 44-49, 2005.

[12] D. Chaum, R. L. Rivest, and A. T. Sherman, "Blind signatures for untraceable payments" *In Advances in Cryptology – Proceedings of. Crypto'82*, pp. 199-203. 1982.

[13] I. B. Damgard. "Payment systems and credential mechanisms with provable security against abuse by individuals," In *Advances in Cryptology - Proceedings of Crypto '88, Lecture Notes in Computer Science*, pp. 328-225, Springer-Verlag, 1988.

[14] D. Chaum, "Blind signature systems," In *Advances in Cryptology - Proceedings of Crypto '83*, pp.153-156, 1983.

[15] N. Ferguson, "Single term off-line coins," In *Advances in Cryptolog - Proceedings of Eurocrypto '93*, pp. 318-328, 1993.

[16] M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party," *Proceedings of The Fourth ACM Conference on Computer and Communications Security*, pp. 1-6, 1997.

[17] M. Franklin and M. Yung, "Secure and efficient off-line digital money," *Proceedings of the twentieth International Colloquium on Automata, Languages and Programming (ICALP 1993), Lecture Notes in Computer Science 700*, pp. 265-276, Springer-Verlag, 1993.

[18] X. S. Hou and C. W. Tan, "A new electronic cash model," *Conference on Information Tech-nology: Coding and Computing (ITCC04)*, Vol.1, pp.374-379, 2004.

[19] B. G. Kim, S. J. Min and K. J. Kim, "Fair tracing based on VSS and blind signature without Trustees," *International research center of Information Security (IRIS) Computer Security Symposium*, 2003.

[20] S. J. Kim and H. K. Oh, "Efficient anonymous cash using the hash chain," *IEICE Transactions on Communications*, Vol.E86-B, No.3, pp. 1140-1143, 2003.

[21] T. Nakanishi and Y. Sugiyama, "An efficient on-line electronic cash with unlikable exact payments," *Proceedings of International Conference on Information Security (ISC2004)*, Vol.3225, pp. 367-378, 2004.

[22] T. Okamoto, "An efficient divisible electronic cash Scheme," In *Advances in Cryptology – Proceedings of Crypto'95, Lecture Notes in Computer Science 963*, pp. 438-451, Springer-Verlag, 1995.

[23] T. Okamoto and K. Ohta, "Universal electronic cash," In *Advances in Cryptology – Proceedings of Crytpo'91*, pp. 324-337, 1998.

[24] T. Okamoto and K. Ohta, "How to simultaneously exchange secrets by general assumption," *Proceedings of 2nd ACM Conference on Computer and Communications Security*, pp. 184-192, 1994.

[25] B. Schoenmakers, "Basic Security of the ecashTM payment system," *Proceedings of Computer Security and Industrial Cryptography: State of the Art and Evolution*, East Course. Leunen Beligium, 1997.

[26] R. Song and L. Korba, "How to make e-cash with non-repudiation and anonymity," *Conference on information Technology: Coding and Computing (ITCC'04),* NRC: 46549, 2004.

[27] M. Stadler, "Publicly verifiable secret sharing," In *Advances in Cryptology – Proceedings of Eurocrypt'96, Lecture Notes in Computer Science*, pp. 190-199, Springer-Verlag, 1996.

[28] Y. Tsiounis, *Efficient electronic cash: new notations and techniques*, Ph.D. Dissertation, College of Computer Science, Northeastern University, Boston, USA, pp.50-51, 1997.

[29] C. H. Wang, "Untraceable fair network payment protocols with off-Line TTP," In *Advances in Cryptology – Proceeings of Asiacrypt'03, Lecture Notes in Computer Science 2894*, pp.173–187, Springer-Verlag, 2003.

[30] H. Wang and Y. Zhang, "Untraceable off-line electronic cash flow," *Proceedings of Austral-Asian E-Commerce Computer Science Conference (ACSC'01)*, Vol.23, No.1, pp. 191-198, 2001.